

AUTOMATED TONE TRANSCRIPTION

Steven Bird

University of Edinburgh, Centre for Cognitive Science
2 Buccleuch Place, Edinburgh, EH8 9LW, UK
Internet: `Steven.Bird@ed.ac.uk`

Abstract

In this paper I report on an investigation into the problem of assigning tones to pitch contours. The proposed model is intended to serve as a tool for phonologists working on instrumentally obtained pitch data from tone languages. Motivation and exemplification for the model is provided by data taken from my fieldwork on Bamileke Dschang (Cameroon). Following recent work by Liberman and others, I provide a parametrised F_0 prediction function \mathcal{P} which generates F_0 values from a tone sequence, and I explore the asymptotic behaviour of downstep. Next, I observe that transcribing a sequence X of pitch (i.e. F_0) values amounts to finding a tone sequence T such that $\mathcal{P}(T) \approx X$. This is a combinatorial optimisation problem, for which two non-deterministic search techniques are provided: a genetic algorithm and a simulated annealing algorithm. Finally, two implementations—one for each technique—are described and then compared using both artificial and real data for sequences of up to 20 tones. These programs can be adapted to other tone languages by adjusting the F_0 prediction function.

INTRODUCTION

The wealth of literature on tone and intonation has amply demonstrated that voice pitch (F_0) in speech is under independent linguistic control. In English, voice pitch alone can signal the distinction between a statement and a question. Similarly, in many tone languages, voice pitch alone signals the tense of a verb. Phonologists usually describe a pitch contour much as they describe speech more generally, namely as a sequence of discrete units (i.e. a transcription). This is illustrated in Figure 1, where L indicates a low tone and \downarrow H indicates a downstepped high tone. The question addressed in this paper concerns how we

should relate pitch contours to tone sequences.

This paper is divided into four main sections, summarised in turn below.

Tone Transcription In this section I present the problem of relating sequences of F_0 values to tone transcriptions. I argue that Hidden Markov Models are unsuited to the task and I demonstrate the importance of having a computational tool which allows phonologists to experiment with F_0 scaling parameters.

F_0 Scaling This section gives a mathematical basis for a general approach to F_0 scaling which, it is hoped, will be applicable to any tone language. I derive an F_0 prediction function from first principles and show how the model of Liberman et al. (1993) for the Nigerian language Igbo is a special case.

Tone and F_0 in Bamileke Dschang Here I present some data from my own fieldwork and give a statistical analysis, using the same technique used by Liberman et al. I then show how the general model of the previous section is instantiated for this language. This demonstrates the versatility of the general model, since it can be applied to two very different tone languages.

Implementations This section provides two non-deterministic techniques for transcribing an F_0 string. The first method uses a genetic algorithm while the second method uses simulated annealing. The performance of both implementations is evaluated and compared on a range of artificial and real data. Finally, I give some examples of multiple, automatically-generated transcriptions of the same F_0 data.

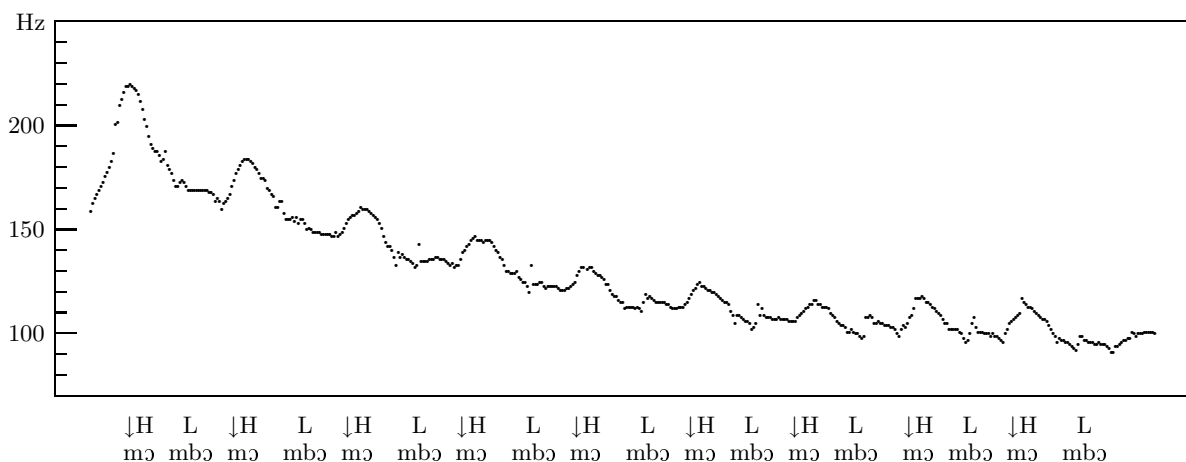


Figure 1: F₀ Trace for Bamileke Dschang Utterance: ‘child and child and ...’

TONE TRANSCRIPTION

Generation and Recognition

A promising way of generating contours from tone sequences is to specify one or more pitch targets per tone and then to interpolate between the targets; the task then becomes one of providing a suitable sequence of targets (Pierrehumbert & Beckman, 1988). It is perhaps less clear how we should go about recognising tone sequences from pitch contours. Hidden Markov Models (HMMs) (Huang et al., 1990) offer a powerful statistical approach to this problem, though it is unclear how they could be used to recognise the units of interest to phonologists. HMMs do not encode timing information in a way that would allow them to output, say, one tone per syllable (or vowel). Moreover, the same section of a pitch contour may correspond to either H or L tones. For example, a H between two Hs looks just like an L between two Ls. There is no principled upper bound on the amount of context that needs to be inspected in order to resolve the ambiguity, leading to a multiplication of state information required by the HMM and problems for training it.

In the present context, the emphasis is not on automatic speech recognition but on a tool to support phonologists working with tone. As we shall see in the next section, once the phonologist has identified the salient location to measure the ‘F₀ value’ of a syllable (or some other phonological unit), the task will be to automatically map a string of these values to a string of tones.

A Tool for Phonologists

Connell and Ladd have devised a set of heuristics for identifying key points in an F₀ contour to record F₀ values (Connell & Ladd, 1990, 21ff). In the absence of a program which enshrines these heuristics, it was decided to develop a system for producing a tone transcription from a sequence of F₀ values. Apart from the obvious benefits of automating the process, such as speed and accuracy, it could show up cases where there is more than one possible tone transcription, possibly with different parameter settings for the F₀ scaling function. Having the *set* of tone transcriptions that are compatible with an utterance has considerable value to an analyst searching for invariances in the tonal assignments to individual morphemes.

To exemplify this point, it is worth considering a recent example where an alternative transcription of some data proved valuable in providing a fresh analysis of the data. In their analyses of tone in Bamileke Dschang, Hyman gives the transcription in (1a) while Stewart gives the one in (1b), for the phrase meaning *machete of dogs*.

- (1) a. ɲɲĩ məm↓bhǎ — (Hyman, 1985, 50)
- b. ɲɲĩ↑' ↓məmbhǎ — (Stewart, 1993, 200)

These two possibilities exist because of different F₀ *scaling parameters*. These parameters determine the way in which the different tones are scaled relative to each other and to the speaker’s pitch range. This is illustrated in (2), adapting Hyman’s earlier notation (Hyman, 1979).

(2) a. Hyman: j̃ɪ̃ m̃ə̃m̃↓bh̃ɛ̃

j̃ɪ̃	j̃ɪ̃	í	m̃ə̃	↓	mbh̃ɛ̃
L	L	H	L	↓	H
3	3	1	3		1
0	0	0	0	1	1
3	3	1	3		2

b. Stewart: j̃ɪ̃j̃ɪ̃↑' ↓m̃ə̃mbh̃ɛ̃

j̃ɪ̃	j̃ɪ̃	↑	í	↓	m̃ə̃	mbh̃ɛ̃
L	L	↑	H	↓	L	H
2	2		1		2	1
1	1	0	0	1	1	1
3	3		1		3	2

Example (2) displays a kind of phonetic interpretation function. Immediately below the two rows of tones we see a row of numbers corresponding to the tones. For Hyman, L=3 and H=1, while for Stewart, L=2 and H=1. Observe in Hyman's example that a rising tone—symbolised by a wedge above the *i*—is modelled as an LH sequence in keeping with standard practice in African tone analysis.

The second row of numbers corresponds to downstep (↓) and upstep (↑). For Hyman's model, this row begins at 0 and is increased by 1 for each downstep encountered. For Stewart's model, this row begins at 1 and is increased by 1 for each downstep encountered and decreased by 1 for each upstep encountered. The two rows are summed vertically to give the last row of numbers. Observe that the last rows of Stewart's and Hyman's models are identical.

The parameter which distinguishes the two approaches is *partial* vs. *total* downstep. Hyman treats Dschang as a *partial downstep language*, i.e. where ↓H appears as a mid tone (with respect to the material to its left). Stewart treats it as a *total downstep language*, i.e. where ↓H appears as an L tone (with respect to the material to its left).

While Hyman and Stewart present rather different analyses of rather different looking transcriptions, we can see that they are really analyzing the same data, given the above interpretation function. Therefore, phonologists who do not wish to limit themselves to the transcriptions which result from certain parameter settings in the phonetic interpretation function would be better off working directly with number sequences like the last row in (2). This paper describes a tool which lets them do just that.

F₀ SCALING

Consider again the F₀ contour in Figure 1. In particular, note that the F₀ decay seems to be to a non-zero asymptote, and that H and L appear to have different asymptotes which we symbolise as *h* and *l* respectively. These observations are clearer in Figure 2, which (roughly speaking) displays the peaks and valleys from Figure 1.

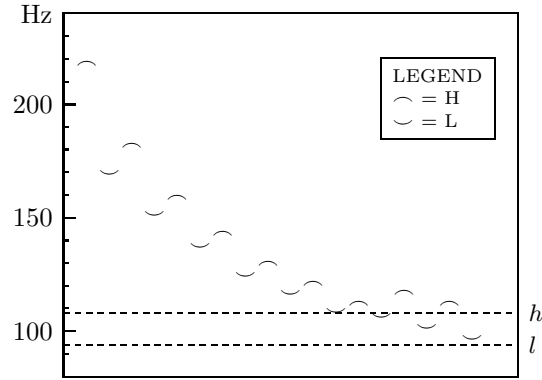


Figure 2: Asymptotic Behaviour of F₀

Although this is admittedly a rather artificial example, it remains true that there is no principled upper limit on the number of downsteps that can occur in an utterance (Clements, 1979, 540), and so the asymptotic behaviour of F₀ scaling still needs to be addressed.

Now suppose that we have a sequence *T* of tones where *t_i* is the *i*th tone (H or L) and a sequence *X* of F₀ values where *x_i* is the F₀ value corresponding to *t_i*. Then we would like a formula which predicts *x_i* given *x_{i-1}*, *t_i* and *t_{i-1}* (*i* > 1). We express this as follows:

$$x_i = \mathcal{P}_{t_{i-1}t_i}(x_{i-1})$$

The question, now, is what should this function look like? Suppose for sake of argument that the ratio of L to the immediately preceding H in Figure 2 is constant, with respect to the baselines for H and L, namely *h* and *l*. Then we have:

$$\frac{x_i - l}{x_{i-1} - h} = c$$

More generally, suppose that we have a sequence of two arbitrary tones. Ignoring the possibility of downstep for the present, we have a static two-tone system where HH and LL sequences are level

and sequences like HLHLHL are realised as simple oscillation between two pitches. We can write the following formula, where $\bar{t}_i = h$ if $t_i = H$ and $\bar{t}_i = l$ if $t_i = L$.

$$\begin{aligned} \frac{x_i - \bar{t}_i}{x_{i-1} - \bar{t}_{i-1}} &= \frac{\bar{t}_i}{\bar{t}_{i-1}} \\ \Rightarrow x_i &= \frac{\bar{t}_i}{\bar{t}_{i-1}} \cdot x_{i-1} \end{aligned}$$

The situation becomes more interesting when we allow for downdrift and downstep. *Downdrift* is the automatic lowering of the second of two H tones when an L intervenes, so HLH is realised as $[-_-]$ rather than as $[-_-]$, while *downstep* is the lowering of the second of two tones when an intervening L is lost, so H↓H is realised as $[-_-]$ (Hyman & Schuh, 1974). Bamileke Dschang has downstep but not downdrift while Igbo has downdrift but only very limited downstep. Now we define $\bar{t}_i = h$ if $t_i = H, \downarrow H$ and $\bar{t}_i = l$ if $t_i = L, \downarrow L$. Generalising our equation once more, we have the following, where R is a factor called the *transition ratio*.

$$\begin{aligned} \frac{x_i - \bar{t}_i}{x_{i-1} - \bar{t}_{i-1}} &= \frac{\bar{t}_i}{\bar{t}_{i-1}} R_{t_{i-1}t_i} \\ \Rightarrow x_i = \mathcal{P}_{t_{i-1}t_i}(x_{i-1}) &= \frac{\bar{t}_i}{\bar{t}_{i-1}} R_{t_{i-1}t_i} \cdot x_{i-1} \\ &+ \bar{t}_i(1 - R_{t_{i-1}t_i}) \end{aligned}$$

Now I shall show how this general equation relates to the equations for Igbo (Liberman et al., 1993, 151), reproduced below:

- (3) HH $x_i = x_{i-1}$
 HL $x_i = (Fl/h)x_{i-1} + l(1 - F)$
 LH $x_i = (h/l)x_{i-1}$
 LL $x_i = Fx_{i-1} + l(1 - F)$
 H↓H $x_i = Dx_{i-1} + h(1 - D)$

\mathcal{P} can be instantiated to the set of equations in (3) by setting R as follows:

t_{i-1}	t_i			
	H	L	↓H	
H, ↓H	1	F	D	$0 < F < 1$
L	1	F	–	$0 < D < 1$

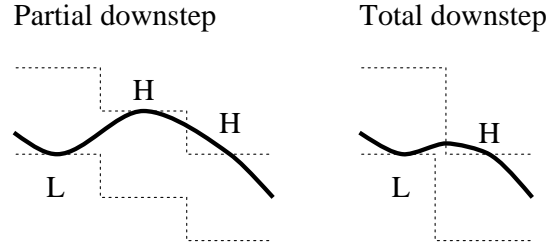
It will be helpful to introduce one more level of generality. \mathcal{P} relates *adjacent* F_0 values, but we would also like to relate *non-adjacent* values, given the sequence of intervening tones. Suppose that $T = t_0 \cdots t_n$ is a tone sequence where the F_0 value of t_0 is x . Then we shall write the F_0 value of t_n as $\mathcal{P}_T(x)$. By repeated applications of \mathcal{P} we

can write down the following expression for \mathcal{P}_T :

$$\mathcal{P}_T(x) = \frac{\bar{t}_n}{\bar{t}_0} R_T \cdot x + \bar{t}_n(1 - R_T)$$

where $R_T = \prod_{k=1}^n R_{t_{k-1}t_k}$, $n > 2$. Now, suppose that $S = s_0 \cdots s_m$ and $T = t_0 \cdots t_n$ are tone sequences and that $\bar{s}_0 = \bar{t}_0$, $\bar{s}_m = \bar{t}_n$ and $\mathcal{P}_S = \mathcal{P}_T$. Then it is straightforward to show that $R_S = R_T$. Notice also that if $\mathcal{P}_T(x) = x$ for all x and if $\bar{t}_0 = \bar{t}_n$ then $R_T = 1$. These results will be useful in the next section.

Finally, it is worth comparing \mathcal{P} with Hyman's and Stewart's interpretation functions which were illustrated in (2). As pointed out already, Hyman's is a partial downstep model while Stewart's is a total downstep model. Partial and total downstep can be visualised as follows, where the dotted lines indicate the abstract *register* inside which tones are scaled, and where downstep corresponds to lowering of the register.



Observe that for partial downstep, it is necessary to have two downsteps before a high tone is at the level of a preceding low, while for total downstep, it is only necessary to have a single downstep for a high tone to be at the same level as the preceding low. We can express these observations about partial and total downstep in the model as follows. For partial downstep, we have $\mathcal{P}_{L\downarrow H\downarrow H}(x) = x$ while for total downstep we have $\mathcal{P}_{L\downarrow H}(x) = x$. For both of these equations we are forced to have $h = l$ which does not seem to be empirically justifiable in view of the data in Figure 1. It might be argued that this indicates a flaw in the model being presented here, since partial and total downstep are widely attested in the literature on tone languages. Unfortunately, it is not possible in general to provide a model for partial or total downstep which permits distinct asymptotes for H and L.¹ Therefore, to the extent that Figure 1 is typical of tone languages in having different H and

¹ To see why this is so for the case of total downstep, suppose that such a model did exist, and so $l < h$. Let $x \in [l, h)$, a valid F_0 value for a low tone. Now, whatever interpretation function \mathcal{P}' we use, we still

L asymptotes, one must conclude that total and partial downstep are qualitative terms only. However, they may yet re-emerge in the model under a different guise, as we shall see later.

The effect of the distinction between partial and total downstep is to allow different transcriptions of the same string, as we saw in (2). In general, we have the following mapping between transcriptions under the two views of downstep:

(4)	partial		total	partial		total
	HH	—	HH	L↓H	—	LH
	HL	—	H↓L	L↓L	—	L↓L
	LH	—	L↑H	H↑H	—	H↑H
	LL	—	LL	H↑L	—	HL
	H↓H	—	H↓H	L↑L	—	L↑L

It is clear that changing from one view of downstep to the other amounts to adding and deleting ↓ and ↑ while leaving the tones themselves unchanged. Thus, the model admits both transcription schemes that result from the two views of downstep, and another besides, as shown later in (7).

This concludes the discussion of the F_0 prediction function. In the next section I shall investigate the phonetic interpretation of tone in Bamileke Dschang, and determine the values of R for this language.

TONE AND F_0 IN BAMILEKE DSCHANG

In a recent field trip to Western Cameroon to study the Bamileke Dschang² noun associative construction, I was able to collect a small amount of data relating to F_0 scaling throughout a particular informant's pitch range. Following Liberman et al., voice pitch was varied by getting the informant to speak at different volumes and by adjusting the recording level appropriately. However, rather than asking the informant to imagine speaking to a subject at different distances, require that $\mathcal{P}'_{L\downarrow H}(x) = x$ by definition of total downstep, which means that there is now a high tone with a F_0 value less than h . But h is the asymptote below which no high tones should ever be realised, and so we have a contradiction. The case for partial downstep follows similarly.

² Bamileke Dschang is a grassfields Bantu language spoken in the Western Province of Cameroon. The name 'Bamileke' (pron: [ba'mileke]) represents both an ethnic grouping and a language cluster; Dschang (pron: [tʃaŋ]) is an important town around which one of the Bamileke languages is spoken. The data here is from the Bafou dialect.

I controlled the volume by having the informant wear headphones and played white noise from a detuned radio. Thus, I could set the informant's voice pitch by using the volume control on my radio. My hypothesis is that this technique produces more consistent volume (and hence, pitch scaling) over long utterances and may make informants less self-conscious about speaking loudly than simply asking them to imagine speaking to subjects at various distances away. Measurements were taken from the following data.

- (5) **HH** á ʒʰó sóŋ té nʒʰó táŋ té nʒʰó túŋ té
 nʒʰó kúp té nʒʰó kóp
*He saw the bird before he saw the hat before
 he saw the basket before he saw the pipe
 before he saw the cup*

LL àpàk — *side, half*

L↓H, HL

è↓só mbò è↓só mbò ... è↓só
jealousy and jealousy and ... jealousy
 là↓pá mbò là↓pá mbò ... là↓pá
breast and breast and ... breast
 mè↓vét mbò mè↓vét mbò ... mè↓vét
oil and oil and ... oil
 ɔ́mó mbò ɔ́mó mbò ... ɔ́mó
child and child and ... child

Regrettably, the LL data was only available from isolated disyllables, and other sequences such as LH and H↓H were not available at all. However, from the F_0 data for the above utterances we can hypothesise the behaviour of these unseen sequences, and this can be tested in subsequent empirical work. The results for utterances involving HH and LL sequences are displayed in Figure 3, while results for L↓H and HL are displayed in Figure 4.

The regression equations obtained from these data are displayed in (6), where the number of occurrences of each tone sequence is given in parentheses after the sequence. The third column gives the standard error for the gradient and intercept.

(6)	Tone Sequence	Regression Equation	Standard Error
	HH (119)	$x_i = 0.99x_{i-1} + 0.91$	0.012, 5.0
	LL (11)	$x_i = 1.02x_{i-1} - 1.39$	0.057, 3.6
	HL (40)	$x_i = 0.65x_{i-1} + 25.0$	0.015, 3.1
	L↓H (38)	$x_i = 1.10x_{i-1} + 0.54$	0.026, 4.3

From this, we conclude that HL is the only sequence with an intercept significantly different from zero, and that $x_i = x_{i-1}$ for HH and LL

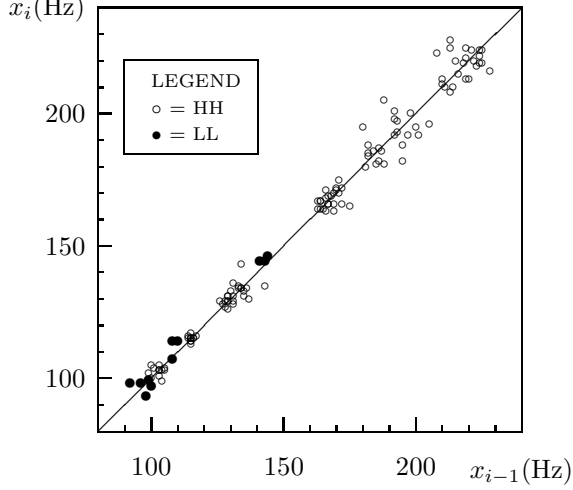


Figure 3: Plot of x_{i-1} vs x_i for HH, LL

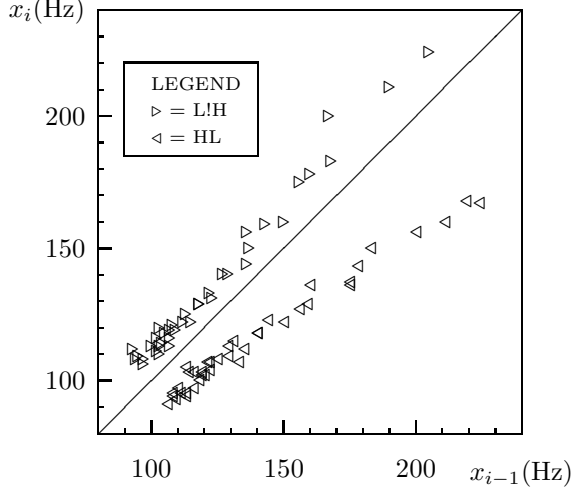


Figure 4: Plot of x_{i-1} vs x_i for L↓H, HL

sequences. We also conclude that $R_{HH} = R_{LL} = R_{L\downarrow H} = 1$, ($l/h = 1.1$) and $R_{HL} = 0.72$. This last value will be referred to as the quantity d . We also see that $l = 88\text{Hz}$ and $h = 96\text{Hz}$. Fortunately, these figures are sufficient to determine the R values for all other pairs of tones in Bamileke Dschang.

A further observation is that Bamileke Dschang does not have downdrift, and so there is no F_0 difference across HLH and LHL sequences. This is evident in Figure 5. Therefore, we can write $\mathcal{P}_{HLH}(x) = x$, and by a result we showed above, $R_{HL} \cdot R_{LH} = 1$. Given that $R_{HL} = d$ it follows that $R_{LH} = \frac{1}{d}$.

Concerning downstep, I shall assume that the magnitude of downstep is independent of the tones

on either side, and so $\mathcal{P}_{HL\downarrow H} = \mathcal{P}_{H\downarrow L} = \mathcal{P}_{L\downarrow L} = \mathcal{P}_{LH\downarrow L}$. A separate instrumental study supports this hypothesis (Bird & Stegen, 1993). Therefore, we have $R_{st} = \frac{1}{d}R_{s\downarrow t} = dR_{s\uparrow t}$, where s is any tone and t is H or L.

Finally, it is important to briefly consider up-step, since it has been used in some analyses of Bamileke Dschang (e.g. Stewart's). Given that up-step and downstep are intended as inverses of each other, we have the identities $\mathcal{P}_{s\downarrow t\uparrow t} = \mathcal{P}_{st} = \mathcal{P}_{s\uparrow t\downarrow t}$, with s, t as before. We now have a complete table for R :

t_{i-1}	t_i					
	H	L	↓H	↓L	↑H	↑L
H, ↓H, ↑H	1	d	d	d^2	d^{-1}	1
L, ↓L, ↑L	d^{-1}	1	1	d	d^{-2}	d^{-1}

Observe the symmetries in this table. The configuration of four R values that we find when t_i is not downstepped or upstepped (the first two columns) is reproduced in the columns for downstep (multiplied by d) and in the columns for upstep (divided by d).

Note also that the above table is dependent upon how the data in (5) was transcribed. Suppose that we had not used repetitions of HL↓H (a transcription scheme based on partial downstep) but H↓LH (a scheme based on total downstep). Then we would have had $R_{H\downarrow L} = d$ and $R_{LH} = 1$. Accordingly, the table for R would be as follows:

t_{i-1}	t_i					
	H	L	↓H	↓L	↑H	↑L
H, ↓H, ↑H	1	1	d	d	d^{-1}	d^{-1}
L, ↓L, ↑L	1	1	d	d	d^{-1}	d^{-1}

The fact that we have two possible tables for R is no cause for alarm. Recall that the transition between two tones t_{i-1} and t_i also involves the factor \bar{t}_i/\bar{t}_{i-1} . This factor is manifested in tone transitions according to the following pattern:

t_{i-1}	t_i					
	H	L	↓H	↓L	↑H	↑L
H, ↓H, ↑H	1	l/h	1	l/h	1	l/h
L, ↓L, ↑L	h/l	1	h/l	1	h/l	1

I therefore conclude that the presence of more than one table for R indicates an interplay between R values and the ratio h/l . This raises an interesting question. Suppose we have two tone sequences $T = t_0 \cdots t_n$ and $T' = t'_0 \cdots t'_n$, and two interpretation functions \mathcal{P} and \mathcal{P}' based

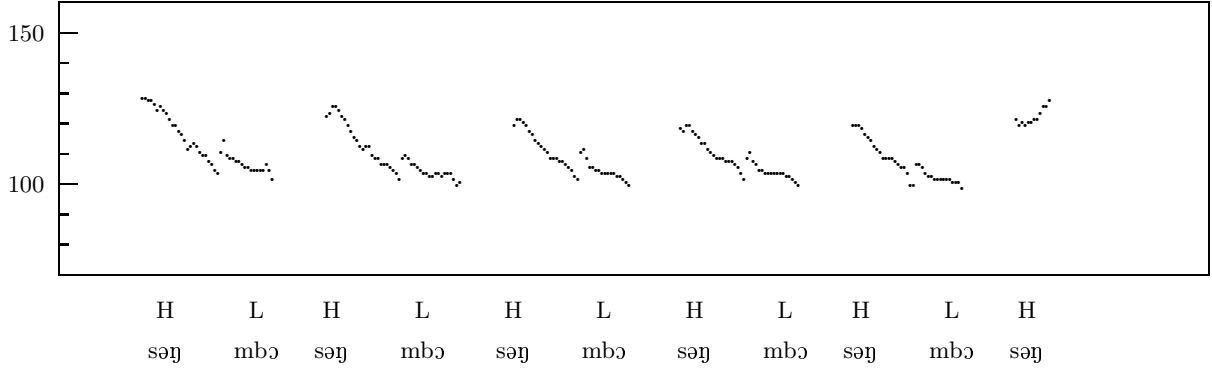


Figure 5: F₀ Trace for ‘bird and bird and ...’

on R and R' respectively. Then under what circumstances is the phonetic interpretation of both sequences the same under their respective interpretation functions? A sufficient condition for them to be the same is that $\bar{t}_i = \bar{t}'_i$ and that $R_{t_{i-1}t_i} = R'_{t'_{i-1}t'_i}$. The reader can check that these conditions are met by the mapping in (4) and the two tables for R given above. Note that this observation holds for the model in general, not just for the specialised version of the model as applied to Bamileke Dschang.

It can also be shown that R is completely determined once R_{HL} is specified. A possible characterisation of total vs. partial downstep now arises: if $R_{HL} = 1$ then we have total downstep, but if $R_{HL} = d < 1$ then we have partial downstep. However, the interpretation of these terms must necessarily be different from the standard interpretation, since I have shown that the standard interpretation is not compatible with the present model.

This concludes the discussion of F₀ scaling in Bamileke Dschang. I shall now present the implementations.

IMPLEMENTATIONS

In this section, I show how it is possible to get two programs to produce a sequence of tones T (i.e. a tone transcription) given a sequence of n F₀ values X . The programs make crucial use of the prediction function \mathcal{P} in evaluating candidate tone transcriptions.

Both programs involve search, and in general, the aim in searching is to discover the values for

x_1, \dots, x_n so as to optimise the value of a specified evaluation function $f(x_1, \dots, x_n)$. When f has many local optima, deterministic methods such as hill-climbing perform poorly. This is because they terminate in a local optimum and the particular one found depends heavily on the starting point in the search, and there is usually no way of choosing a good starting point.

Exhaustive search for the global optimum is not an option when the search space is prohibitively large. In the present context, say for a sequence of 20 tones, the search space contains $6^{20} \approx 10^{15}$ possible tone transcriptions, and for each of these there are thousands of possible parameter settings, too large a search space for exhaustive search in a reasonable amount of computation time.

Non-deterministic search methods have been devised as a way of tackling large-scale combinatorial optimisation problems, problems that involve finding optima of functions of discrete variables. These methods are only designed to yield an approximate solution, but they do so in a reasonable amount of computation time. The best known such methods are genetic search (Goldberg, 1989) and annealing search (van Laarhoven & Aarts, 1987). Recently, annealing search has been successfully applied to the learning of phonological constraints expressed as finite-state automata (Ellison, 1992). In the following sections I describe a genetic algorithm and an annealing algorithm for the tone transcription problem.

A Genetic Algorithm

For a cogent introduction to genetic search and an explanation of why it works, the reader is referred to (South et al., 1993). Before presenting the version of the algorithm used in the implementation, I shall informally define the key data types it uses along with the standard operations on those types.

gene A linear encoding of a solution. In the present setting, it is an array of n tones, where each tone is one of H, ↓H, ↑H, L, ↓L or ↑L. A gene also contains 16 bit encodings of the parameters h , l and d . These encodings were scaled to be floating point numbers in the range [90, 110] for h , [70, 100] for l and [0.6, 0.9] for d .

gene pool An array of genes, P . One of the search parameters is the size of P , known as the *population*. The gene pool is renewed each generation, and the number of generations is another search parameter.

evaluation A measure of the fitness of a gene as a solution to the problem. Suppose that X is the sequence of F_0 values we wish to transcribe. Suppose also that T is a particular gene. The the evaluation function is as follows:

$$\mathcal{E}_X(T) = \frac{1}{n} \sum_{i=2}^n (\mathcal{P}_{t_{i-1}t_i}(x_{i-1}) - x_i)^2$$

crossover This is an operation which takes two genes and produces a single gene as the result. Suppose that $A = a_1 \cdots a_n$ and $B = b_1 \cdots b_n$. Then the crossover function C_r is defined as follows, where r is the (randomly selected) *crossover point* ($0 \leq r \leq n$).

$$\begin{aligned} C_r(a_1 \cdots a_r a_{r+1} \cdots a_n, b_1 \cdots b_r b_{r+1} \cdots b_n) \\ = a_1 \cdots a_r b_{r+1} \cdots b_n \end{aligned}$$

In other words, the genes A and B are cut at a position determined by r and the first part of A is spliced with the second part of B to create a new gene. Crossover builds in the idea that good genes tend to produce good offspring. To see why this is so, suppose that the transcription contained in the first part of A is relatively good while the rest is poor, while the transcription contained in the first part of B is poor and the rest is relatively good. Then the offspring containing the first part of A and the second part of B will be an improvement on both A and B ; other possible offspring from A and B

will be significantly worse and may not survive to the next generation. The program performs this kind of crossover for the parameters h , l and d , employing independent crossover points for each, and randomising the argument order in C_r so that the high order bits in the offspring are equally likely to come from either parent.

An extension to crossover allows more than one crossing point. The current model permits an *arbitrary number of crossing points* for crossover on the transcription string. The resulting gene is optimal since we choose the crossing points in such a way as to minimise $(\mathcal{P}_{t_{i-1}t_i}(x_{i-1}) - x_i)^2$ at each position. In developing the system, exploiting the decomposability of the evaluation function in this way caused a significant improvement in system performance over the version which used simple crossover.

breeding For each generation, we create a new gene pool from the previous one. Each new gene is created by mating the best of three randomly chosen genes with the best of three other randomly chosen genes.

mutation In order to maintain some *genetic diversity* and an element of randomness throughout the search (rather than just in the initial configuration), a further operation is applied to each gene in every generation. With a certain probability (known as the *mutation probability*), for each gene T and each tone in T , the tone is randomly set to any of the six possible tones. Likewise, the parameter encodings are mutated. The mutation rate is set to 0.005 but raised to 0.5 for a single generation if the evaluation of the best gene is no improvement on the evaluation of the best gene ten generations earlier. The best gene is never mutated.

The building blocks of genetic search discussed above are structured into the following algorithm, expressed in pseudo-Pascal:

procedure genetic_search

begin

initialise Pool, NewPool;

 for g := 1 to generations do

 begin

 if **good_performance**(10) then

 mutation_rate := 0.005;

 else

 mutation_rate := 0.5;

 NewPool[1] := **find_best_gene**(Pool);

 for n := 2 to population do


```

begin
  gene1 := best_of_three(Pool);
  gene2 := best_of_three(Pool);
  NewPool[n] := crossover(gene1, gene2);
  mutate(NewPool[n], mutation_rate);
end
Pool := NewPool;
evaluate(Pool);
end
write find_best_gene(Pool);
end

```

The main loop is executed for each generation. Each time through this loop, the program checks performance over the last ten generations and if performance has been good, the mutation rate stays low, otherwise it is changed to high. Then it copies the best gene to the new pool. Now we reach the inner loop, which selects two genes, performs crossover, and mutates the result. Next, the current pool is updated, an evaluation is performed, and the program continues with the next generation. Once all the generations have been completed, the program displays the best gene from the final population and terminates.

An Annealing Algorithm

As with genetic algorithms, simulated annealing (van Laarhoven & Aarts, 1987) is a combinatorial optimisation technique based on an analogy with a natural process. *Annealing* is the heating and slow cooling of a solid which allows the formation of regular crystalline structure having a minimum of excess energy. In its early stages when the temperature is high, annealing search resembles random search. There is so much free energy in the system that a transition to a higher energy state is highly probable. As the temperature decreases the search begins to resemble hill-climbing. Now there is much less free energy and so transitions to higher energy states are less and less likely. In what follows, I explain some of the parameters of annealing search as used in the current implementation.

temperature At the start of the search the temperature, t is set to 1. During the search, the temperature is reduced at a rate set by the ‘cooling rate’ parameter, until it reaches a value less than 10^{-6} .

perturbation At each step of the search, the current state is perturbed by an amount which depends on the temperature. The temperature determines the fraction of the search space that

is covered by a single perturbation step. For a tone sequence of length n , we randomly reset the worst $n.t$ tones according to $(\mathcal{P}_{t_{i-1}t_i}(x_{i-1}) - x_i)^2$. For the parameters we proceed as follows, here exemplified for h . First, set $\rho = t(h_{\max} - h_{\min})$. Now, add to h a random number in the range $[-\rho, \rho]$ and check that the result is still in the range $[h_{\min}, h_{\max}]$.

equilibrium At each temperature, the system is required to reach ‘thermal equilibrium’ before the temperature is lowered. In the present context, equilibrium is reached if no more than one of the last eight perturbations yielded a new state that was accepted.

free energy function This is the amount of available energy for transitions to higher energy states. In the current system, it is the distribution $-1000.t.\log(p)$, where p is a uniform random variable in the range $(0, 1]$. If the energy difference Δ between an old and a new state is less than the available energy, then the transition is accepted. The factor of 1000 is intended to scale the energy distribution to typical values of the evaluation function.

Now the algorithm itself is presented:

procedure annealing_search

```

begin
  initialise Trans, NewTrans, BestTrans;
  randomise Trans;
  t := 1;
  while t > 0.000001 do
    begin
      repeat
        NewTrans := perturb(Trans, t);
         $\Delta$  := evaluate(NewTrans)
          - evaluate(Trans);
        if  $\Delta < 0$  or
           $\exp(-\Delta/1000.t) > \text{random}(0,1)$  then
          Trans := NewTrans;
        if evaluate(Trans) < evaluate(BestTrans)
          BestTrans := Trans;
      until equilibrium_reached;
      Trans := BestTrans;
      temperature := temperature / 1.2;
    end
    write Trans;
  end
end

```

The program is made up of two loops. The outer loop simply iterates through the temperature range, beginning with a temperature of 1 and steadily decreasing it until it gets very close to

zero. The nested loop performs the task of reaching thermal equilibrium at each temperature. The first step is to perturb the previous transcription to make a new one. Notice that the temperature t is a parameter of the perturb function. Next, the difference Δ between the old and new evaluations is calculated. If the new transcription has a better evaluation than the old one, then Δ is negative. Next, the program accepts the new transcription if (i) Δ is negative or (ii) Δ is positive and there is sufficient free energy in the system to allow the worse transcription to be accepted. Finally, we check if the new transcription is better than the best transcription found so far (BestTrans) and if so, we set BestTrans to be the new transcription. Once equilibrium is reached, the current transcription is set to be the best transcription found so far, and the search continues.

Performance Results

Both the genetic and annealing search algorithms have been implemented in C++. In this section, the performance of the two implementations is compared. Performance statistics are based on 1,200 executions of each program. Search parameters were set so that each execution took around 5 seconds on a Sun Sparc 10. Three performance trials were undertaken.

Trial 1: Artificial Data. In the first trial, both programs generated random sequences of tones, then computed the corresponding F_0 sequence using \mathcal{P} , then set about transcribing the F_0 sequence. Since these sequences were ideal, the best possible evaluation for a transcription was zero. The performance of the programs could then be measured to see how close they came to finding the optimal solution. Each program was tested on F_0 sequences of length 5, 10, 15 and 20. For each length, each program transcribed 100 randomly-generated sequences. The results are displayed in Figure 6. Each pair of bars corresponds to a given transcription length. The left member of each pair is for the genetic search program, while the right member is for the annealing search program.

The heavily shaded bars corresponding to evaluations less than 1 are the most important. These indicate the number of times out of 100 that the programs found a transcription with an evaluation less than 1. This evaluation means that the average of the squared difference between the predicted F_0 values and the actual F_0 values was less than 1Hz. Observe that the annealing search program performs significantly better in all cases.

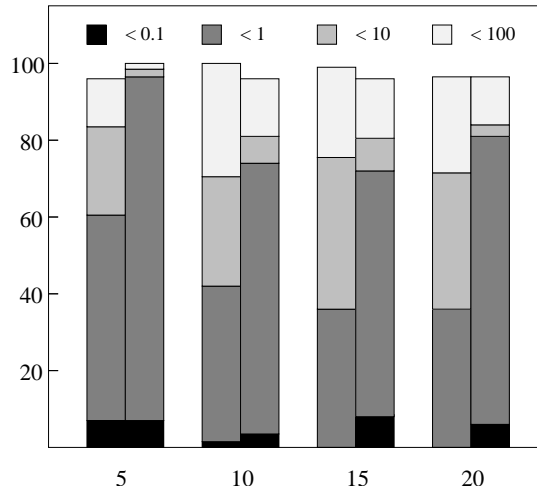


Figure 6: Performance results (no upstep)

Note that the mutation operation in the genetic search program treats each bit in the parameter encodings equally, while the perturbation operation in the annealing search program is sensitive to the distinction between more significant vs. less significant bits. This may explain the better convergence behaviour of the annealing search.

Notice also in Figure 6 that performance does not degrade with transcription length as the length doubles from 10 to 20. This is probably because a randomly generated sequence will contain downsteps on every second tone (on average) causing a general downtrend in the F_0 values and severely limiting the combinatorial explosion of possible transcriptions.

Trial 2: Artificial Data with Upstep. Trial 2 was the same as trial 1 except that this time upstep was permitted as well. The results are displayed in Figure 7. Again the annealing program fares better than the genetic program. Consider again the bars corresponding to evaluations less than 1. For both programs, however, observe that the performance degrades more uniformly than in trial 1, probably because the inclusion of upstep greatly increases the number of possible transcriptions (and hence, the number of local optima).

Trial 3: Actual Data. The final trial involved real data, including data from the utterance given in Figure 1. This trial involved four subtrials. The first and second had F_0 sequences of length 10, while the third and fourth had length 18 and 19. The first and second sequences were taken by extracting the initial 10 F_0 values from the third and fourth sequences, thereby avoiding the asymptotic

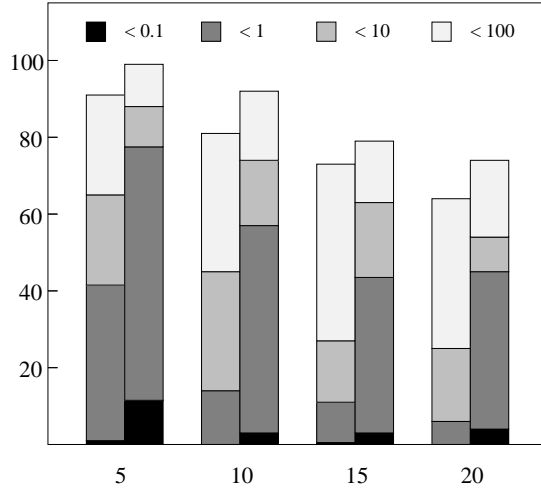


Figure 7: Performance results (upstep)

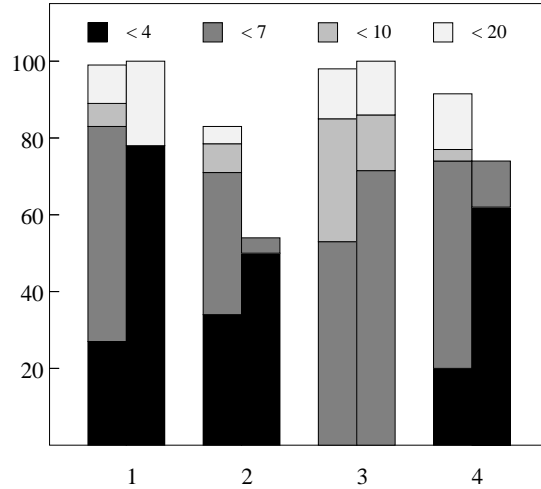


Figure 8: Performance results for actual data

behaviour of the longer sequences. The data is tabulated below, and it comes from the sentences in (5).

Trial	F_0 sequence
1	219,168,183,150,160,136,144,123,131,115
2	205,224,167,200,156,175,136,156,127,140
3	219,168,183,150,160,136,144,123,131,115, 122,107,113,105,118,100,113,95
4	205,224,167,200,156,175,136,156,127,140, 118,129,109,119,103,120,102,111,95

Performance results are given in Figure 8. Notice that the interpretation of the shading in this figure is different from that in previous figures. This is because evaluations near zero were less likely with real data. In fact, the annealing program never found an evaluation less than 3 while the genetic program never found an evaluation less than 4.

Since the programs performed about equally on finding transcriptions with an evaluation less than 7, I shall display these transcriptions along with an indication of how many times each program found the transcription (G = genetic, A = annealing). I give transcriptions which occurred at least twice in one of the programs, during 100 executions of each.

Trial 1: Transcriptions G A
H↓L↓H↓L↓H↓L↓H↓L↓H↓L 27 37
H↓L↓H↓L↓HL↓H↓L↓HL 7 0
H↓L↓HL↓HL↓HL↓HL 3 0
H↓LH↓LH↓LH↓LH↓L 20 2
HL↓HL↓HL↓HL↓HL 24 39

Trial 2: Transcriptions G A
L↓H↓LH↓L↓H↓L↓H↓L↓H 5 0
L↓H↓LH↓L↓H↓LH↓L↓H 66 54

Trial 3: Transcriptions G A
H↓L↓H↓L↓HL↓H↓L↓HLH↓LH↓L 11 0
H↓L↓HL↓HL↓HL↓HL↓H↓LH↓LH↓L 1 2
H↓L↓HL↓HL↓HL↓HL↓HL↓HLH↓LH↓L 10 14
HL↓HL↓HL↓HL↓HL↓HL↓HLH↓LH↓L 30 56

Trial 4: Transcriptions G A
L↓H↓LH↓L↓H↓LH↓L↓H↓L↓H↓LH↓LH↓L 60 29
L↓H↓LH↓L↓H↓LH↓L↓HL↓H↓L↓H↓LH↓LH↓L 5 19
L↓H↓LH↓L↓H↓LH↓L↓HL↓H↓L↓HLH↓LH↓L 7 7
L↓H↓LH↓L↓H↓LH↓L↓H↓L↓H↓LH↓LH↓LH↓L 0 4
L↓H↓LH↓L↓H↓LH↓L↓HL↓HL↓H↓LH↓LH↓L 0 3
L↓H↓LH↓L↓H↓LH↓L↓HL↓HL↓HLH↓LH↓L 0 6

The results from trial 1 deserve special attention. In trial 1, three transcriptions were found by both programs. The best evaluations found are given below:

H L ↓H L ↓H L ↓H L	$\mathcal{E}: \ni h: 107 l: 100 d: 0.68$
H↓L H↓L H↓L H↓L H↓L	$\mathcal{E}: \triangle h: 90 l: 93 d: 0.76$
H↓L↓H↓L↓H↓L↓H↓L↓H↓L	$\mathcal{E}: \ni h: 107 l: 100 d: 0.82$

It is striking to note that the first two transcriptions above are what Hyman and Stewart (respectively) would have given as transcriptions for the abstract F_0 sequence 1 3 2 4 3 5 4 6 5 7. This is demonstrated in (7a,b). The third transcription points to another possibility, given in (7c).

(7) a. **Hyman's transcription scheme**

H	L	↓H	L	↓H	L	↓H	L	↓H	L
1	3	1	3	1	3	1	3	1	3
0	0	1	1	2	2	3	3	4	4
1	3	2	4	3	5	4	6	5	7

b. **Stewart's transcription scheme**

H	↓L	H	↓L	H	↓L	H	↓L	H	↓L
1	2	1	2	1	2	1	2	1	2
0	1	1	2	2	3	3	4	4	5
1	3	2	4	3	5	4	6	5	7

c. **Novel transcription scheme**

H	↓L	↓H	↓L	↓H	↓L	↓H	↓L	↓H	↓L
1	$\frac{5}{2}$	1	$\frac{5}{2}$	1	$\frac{5}{2}$	1	$\frac{5}{2}$	1	$\frac{5}{2}$
0	$\frac{1}{2}$	1	$\frac{3}{2}$	2	$\frac{5}{2}$	3	$\frac{7}{2}$	4	$\frac{9}{2}$
1	3	2	4	3	5	4	6	5	7

Therefore, there are encouraging signs that the program is living up to its promise of producing alternative, equally acceptable transcriptions, as desired from an analytical standpoint.

Multiple Solutions

Although we have seen more than one transcription for a given F_0 sequence, it is inconvenient to be required to run the programs several times in order to see if more than one solution can be found. Furthermore, the programs are designed not to get caught in local optima, which is a problem since interesting alternative transcriptions may actually be local optima. Therefore, both programs are set up to report the k best solutions, where the user specifies the number of solutions desired. The program ensures that the same area of the search space is not re-explored by subsequent searches. This is done by defining a distance metric on transcriptions which counts the number of tones in one transcription that have to be changed in order to make it identical to the other transcription. That part of the search space within a distance of $n/3$ from any previously found solution is not explored again. The programs give up before finding k solutions if 5 randomly generated transcriptions all fall within distance $n/3$ of previous solutions.

Now, consider the following randomly generated sequence of tones:

↑H	↑H	↓H	L	↓L	↑H	L	$h: 107$	$l: 98$
201	215	201	173	163	201	173	$d: 0.87$	$\mathcal{E}: \iota$

The annealing program was set the task of finding ten transcriptions of this tone sequence. The program was run only twice, and it reported the following solutions with evaluations less than or equal to 1. Both runnings of the program found the same solutions, and in the same order. (Note that two transcriptions are taken to be the same if one or both begin with an initial upstep or downstep; this has no effect on the phonetic interpretation). In the following displays, the predicted F_0 values are given below each solution to facilitate comparison with the input sequence.

↓H	↑H	↓H	L	↓L	↑H	L	$h: 101$	$l: 92$
201	215	201	172	163	201	172	$d: 0.88$	$\mathcal{E}: \iota, \in \iota$
↓H	↑H	↓H	↑L	↓L	H	↑L	$h: 109$	$l: 94$
201	215	201	174	163	201	174	$d: 0.87$	$\mathcal{E}: \iota, \in \exists$
L	↓H	↓H	L	↓L	↑H	L	$h: 105$	$l: 97$
201	217	201	174	163	201	174	$d: 0.86$	$\mathcal{E}: \infty, //$

H	↑H	↓H	L	↓L	↑H	L	$h: 110$	$l: 100$
201	214	201	173	164	201	173	$d: 0.88$	$\mathcal{E}: \iota, \forall /$
↑H	↑H	↓H	↑L	↓L	H	↑L	$h: 102$	$l: 88$
201	215	201	174	164	201	174	$d: 0.88$	$\mathcal{E}: \iota, /$
↓L	↓H	↓H	L	↓L	↑H	L	$h: 104$	$l: 96$
201	217	201	174	163	201	174	$d: 0.86$	$\mathcal{E}: \infty, //$

Since all executions to this point have been based on the first table of R values, it was decided to try a test with the second table of R values to see if the performance was different. Interestingly, the third solution in both of the above executions was not found, though two new solutions were found.

↑H	↑H	↓H	L	↓L	↑H	L	$h: 94$	$l: 80$
201	216	201	173	162	201	173	$d: 0.88$	$\mathcal{E}: \iota, \Delta \exists$
L	↓H	↓H	L	↓L	↑H	L	$h: 97$	$l: 84$
201	215	201	174	163	201	174	$d: 0.88$	$\mathcal{E}: \iota, \nabla$
↓H	↑H	↓H	↑L	↓L	H	↑L	$h: 100$	$l: 81$
201	215	201	174	163	201	174	$d: 0.88$	$\mathcal{E}: \iota, \exists \in$
↑L	H	L	↓H	L	↑L	↓H	$h: 92$	$l: 86$
201	214	201	173	163	201	173	$d: 0.67$	$\mathcal{E}: \iota, \Delta \forall$

↓H	↑H	↓H	L	↓L	↑H	L	$h: 107$	$l: 92$
201	216	201	173	162	201	173	$d: 0.87$	$\mathcal{E}: \iota, \Delta \iota$
L	H	L	↓H	L	↑L	↓H	$h: 99$	$l: 93$
201	214	201	173	163	201	173	$d: 0.65$	$\mathcal{E}: \iota, \forall \in$
↓L	↓H	↓H	L	↓L	↑H	L	$h: 90$	$l: 78$
201	217	202	174	163	202	174	$d: 0.88$	$\mathcal{E}: \iota, \forall /$

Observe that the value of d in the above solutions clusters around 0.66 and 0.87. Similar clustering may be occurring with the ratio h/l . However, an analysis of the relationship between the kinds of solutions found, the two R tables and the parameter values h , l and d has not been attempted.

Areas for Further Improvement

It is rather unsatisfying that the performance of the two programs is heavily dependent on the setting of several search parameters, and it seems to be a combinatorial optimisation problem in itself to find good parameter settings. My trial-and-error approach will not necessarily have found optimal parameter values, and so it would be premature to conclude from the performance comparison that annealing search is better than genetic

search for the problem of tone transcription. A more thoroughgoing comparison of these two approaches to the problem needs to be undertaken.

Since the parameters are continuous variables, and since the evaluation function—which we could write as $\mathcal{E}_{T,X}(h,l,d)$ —is a smoothly continuous function in h , l , d , it would be worthwhile to try other (deterministic) search methods for optimising h , l and d , once a candidate tone transcription T has been found.

Finally, it would be interesting to integrate a system like either of the ones presented here into a speech workstation. As the phonologist identifies salient points with a cursor the system would do the transcription, incrementally and interactively.

CONCLUSION

This paper began with a discussion of the problem of relating tone transcriptions to their physical counterparts, namely F_0 traces. I showed that it is desirable for phonologists working on tone to use sequences of F_0 values as their primary data, rather than impressionistic transcriptions which make (usually implicit) assumptions about F_0 scaling. I provided an F_0 prediction function \mathcal{P} which estimated the F_0 value of a tone, given the F_0 value of the previous tone and the identities of the two tones. I presented instrumental data from Bamileke Dschang and showed how the function could be specialised for this language. The function was then incorporated into the evaluation functions of two implemented non-deterministic search algorithms. The performance results were encouraging and demonstrate the promise of automated tone transcription.

ACKNOWLEDGEMENTS

This research is funded by the UK Economic and Social Research Council, under grant R00023 4439 *A Computational Model for the Phonology-Phonetics Interface in Tone Languages*. I am indebted to SIL Cameroon for their logistical support on my field trip in September and October of 1993, during which the data presented in the paper (and much other data besides) was gathered, and especially to Nancy Haynes, Gretchen Harro for helping me collect the data and Jean-Claude Gnintedem who endured many recording sessions. I am grateful to John Coleman, Michael Gasser and Marie South for helpful comments on an earlier version of this paper. The F_0 data was extracted using the ESPS Waves+ package in the

Edinburgh University Phonetics Laboratory.

References

- Bird, S. & Stegen, O. (1993). Tone in the Bamileke Dschang Associative Construction: An Electrolaryngographic Study and Comparison with Hyman (1985). RP 57, University of Edinburgh, Centre for Cognitive Science.
- Clements, G. N. (1979). The description of terraced-level tone languages. *Language*, 55, 536–558.
- Connell, B. & Ladd, D. R. (1990). Aspects of pitch realisation in Yoruba. *Phonology*, 7, 1–29.
- Ellison, T. M. (1992). *Machine Learning of Phonological Structure*. PhD thesis, University of Western Australia.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
- Huang, X. D., Ariki, Y., & Jack, M. (1990). *Hidden Markov Models for Speech Recognition*. Edinburgh Information Technology Series. Edinburgh University Press.
- Hyman, L. M. (1979). A reanalysis of tonal downstep. *Journal of African Languages and Linguistics*, 1, 9–29.
- Hyman, L. M. (1985). Word domains and downstep in Bamileke-Dschang. *Phonology Yearbook*, 2, 45–83.
- Hyman, L. M. & Schuh, R. G. (1974). Universals of tone rules: evidence from West Africa. *Linguistic Inquiry*, 5, 81–115.
- Liberman, M., Schultz, J. M., Hong, S., & Okeke, V. (1993). The Phonetic Interpretation of Tone in Igbo. *Phonetica*, 50, 147–160.
- Pierrehumbert, J. & Beckman, M. (1988). *Japanese Tone Structure*. Cambridge Mass.: MIT Press.
- South, M. C., Wetherill, G. B., & Tham, M. T. (1993). Hitch-hiker's guide to genetic algorithms. *Journal of Applied Statistics*, 20, 153–175.
- Stewart, J. M. (1993). Dschang and Ebrié as Akan-type total downstep languages. In H. van der Hulst & K. Snider (Eds.), *The Phonology of Tone – The Representation of Tonal Register* (pp. 185–244). Berlin; New York: Mouton de Gruyter. Linguistic models, Volume 17.

van Laarhoven, P. J. M. & Aarts, E. H. L. (1987).
Simulated Annealing. Dordrecht:Reidel.